A Parallel Ada Translation System for Massively Parallel Computers[1]
Final Report

Richard A. Volz
Ronald Theriault
Gary Smith
Raymond Waldrop

Texas A&M University
College Station, Texas

1 Introduction

Over the past year, we have, in cooperation with NASA, developed a concept for the distribution of Ada 9X programs and a graphical tool which aids that distribution. The tool, referred to as NASA Distributed Ada Configuration Tool (NDACT) is in a prototype state and has been successfully used to demonstrate the distribution of Ada programs.

As most of the results of this grant are contained in papers, code to be delivered, documentation manuals and support provided to the NYU GNAT team, this report will only highlight the results and will reference attachments for details.

2 Description of Tool

NDACT supports the distribution of Ada 9X programs following the conventions in the distribution annex of the Ada 9X language reference manual[3]. Because of current compiler limitations, however, we are limited to only those features that can be implemented within the confines of Ada 83.

In the following sections, we give a brief overview of some of the major components of NDACT: support for catagorization pragmas, the GUI, the availability layer and support for parallel programming.

2.1 Distribution in Ada 9X

Distribution of Ada 9X programs is accomplished through use of catagorization pragmas and a partition communication system specified in the Distribution Annex of the language and an implementation defined configuration process which is outside the scope of the language. See

[1]This work was supported under NASA contract number NCC-2-791

[2] and [3] for a discussion of the pragmas and the communication system specifications. We only give the briefest introduction to the concepts here.

Essentially, a program is divided into a collection of active and passive partitions. Passive partitions are intended to share memory with those units calling them, while active partitions may be located on distributed processors and subprogram calls made among them. In order for subprograms to be called remotely (without great implementation difficulties), the packages in which they are defined must satisfy certain restrictions. The catagorization pragmas of Ada 9X indicate appropriate restrictions. Partitions are formed as collections of restricted kinds of packages and subprograms satisfying the constraints specified by the categorization pragmas.

The configuration process allows a user to specify the composition of the partitions and the processors to which they are assigned.

2.2 Categorization Pragmas

Distribution in Ada 9X is facilitated by the inclusion of catagorization pragmas within Ada packages. NDACT provides direct support for two of these: Pragma Pure and Pragma Remote_Call_Interface (RCI). When one these pragmas is applied to a package, the package is referred to as a Pure or RCI package, respectively. Pure packages are primarily used to declare type information while RCI packages are used to define the call interface into a partition.

In order for a package to be catagorized as a Pure or RCI package, it must meet a set of restrictions. For example, Pure packages can have no variable state. NDACT uses the Ada Semantic Interface Specification (ASIS) bindings to obtain the information on a program unit which allows it to verify that the specified packages meet the restriction requirements that are applicable to them.

2.3 NDACT Operation and Graphical User Interface

NDACT may either be used as a design tool to specify distribution of a program's units before they is written (and then check afterwards) or take an existing program and effect a distribution of it. The user interacts with NDACT through a convenient to use, Motif-based, GUI.

The first step will usually be to scan in the contents of an Ada library or to load in information from a previous execution of the tool. The Ada units will appear as icons in a window of the GUI. The tool can automatically classify the Ada packages according to the conventions in the distribution annex (described above) or the user can bypass the automatic classification of packages and explicitly catagorize them individually. Catagorization of packages is accomplished through the tool rather than through a pragma because of our current Ada '83 limitations. If the user explicitly catagorizes the packages, the GUI can then be used to verify that those packages satisfy the associated restrictions. In either case, if the package does not meet the restrictions, its icon will move to an error window and the user can view the errors by selecting the icon.

The user will then create a number of partitions. The partitions will also appear as icons and the user can assign RCI packages to those partitions by selecting the RCI package with the mouse and dropping it on the partition. Support for the replication of partitions (described below) is also provided.

The next step will be to assign partitions to actual machines and to have the tool automatically insert communication routines and build executable binary images. The binaries will include any necessary network communication routines. The tool can then be used to run the distributed program.

2.4 Availability Layer

Runtime support for communication between distributed nodes was initially provided by a TeleSoft product, TeleAdaLAN. TeleAdaLAN uses the UDP protocol for message delivery. Because UDP does not guarantee message delivery, TeleAdaLAN provides the ability to retry a message. This, however, can lead to redundant calls. To overcome these problems, we replaced the communication system with a connection-based mechanism and incorporated an availability layer[7] which provides an exception back to the calling routine if the target is inaccessible. The availability layer initially consisted of a tree-based heartbeat mechanism. Recent work has replaced the tree topology with a star topology for greater efficiency.

2.5 Parallel Programming

We have also studied the feasibility of partition replication as an aid to parallel programming. NDACT supports partition replication and we have used this technique in a successful demonstration program (a discussion of this demonstration program is attached).

With NDACT, multiple copies of a partition can be created and assigned to different machines. In order to allow the user program to specify which replicant is desired in a remote call, an extra parameter is added to each subroutine in a replicable RCI package (replicable RCI packages are used to build replicated partitions). The communication system uses this extra parameter as a replicant index and routes the call accordingly.

3 Achievements Resulting from this Project

This project has been an important link in a substantial chain of work on distribution of Ada programs which has led to a series of publications (See reference list for some of the more recent ones). The work supported in this project directly contributed to one of these, "Distributed and Parallel Ada and the Ada 9X Recommendations"[2]. Another paper describing, in detail, the techniques developed under this grant has been submitted to the Joint Workshop on Parallel and Distributed Real-Time Systems[1].

One particularly important aspect of the project has been the opportunity to begin to provide some assistance to the GNAT Ada 9X project and point toward an ultimate availability of a Distribution Annex for it. Under the auspices of this project, assistance has been provided in two primary areas, catagorization pragma checks and stub creation.

In order to assist with the development of catagorization pragma support within GNAT, we provided NYU with a copy of our own catagorization pragma checks. Also, during the development of our own checks, we had developed a set of test cases. These test cases were later used to provide early feedback and error reports to NYU on their catagorization pragma support.

We have also provided information on stub creation and streams to NYU. NDACT provides automatic stub creation and parameter flattening routines (related to 9X streams) and the same type of routines will be needed within GNAT. Drawing from our experience, we were able to make suggestions to NYU on their Stream_IO strategies and developed a document describing stub creation.

A description of a demonstration program which we have used on several occasions can be found in AppendixA.

4 Future Work

We have recently focused our attention on supporting Ada 9X distribution with the GNAT compiler. During the course of this project we have provided support to NYU for their implementation of the Distribution Annex and have recently entered into a joint agreement with CSC to providing a working implementation of Ada 9X distribution, using GNAT as a target compiler.

We are also investigating the possibility of providing results from our project on a publicly available WWW page. As noted above, we are also writing a paper describing our work.

5 Attached Documentation

Attached you will find four documents: a user manual, a design document, and a copy of the two papers resultant from this work[2, 1]. A tape containing source code created will be submitted under separate cover.

A Mandlebrot Demonstration Program

A demonstration program we have used on several occasions is a distributed calculation of a Mandlebrot image. The program calculates and displays the same image in two different windows. The entire image in one window is calculated by a single processor. The image in the other window is calculated by two or more processors. A screen capture of the program is shown in Figure 1.

The distributed program consists of a display partition and a replicable image calculation partition. The display partition contains the main program and an RCI package to define the interface into the partition. The RCI package includes procedures to create an image window and a procedure to display a single line of image data.

The interface to the image calculation partition is specified by a replicable RCI package. The replicable RCI includes a procedure to calculate a portion of the Mandlebrot image. Because it is replicable, all subprograms defined in the RCI specification contain a Partition_Index parameter. The Partition_Index is used to distinguish between instances of the replicable partition. A pure package is included in both partitions for type sharing.

We use three (or more) replicants of the image calculation partition. The first replicant is used to calculate the entire image in one window. At least two other replicants are used to calculate the exact same image in the other window. The display partition assigns work to each of the replicants. The replicants then calculate and send lines of image data back to the display partition.

Each of the partitions is configured onto a different processor. By showing the windows side by side we demonstrate that the image calculated by two processors is finished before the image calculated by a single processor. In addition, we can vary the number of replicants at run time. Additional replicants cause the image on the right to be divided into smaller segments with a corresponding decrease in calculation time.

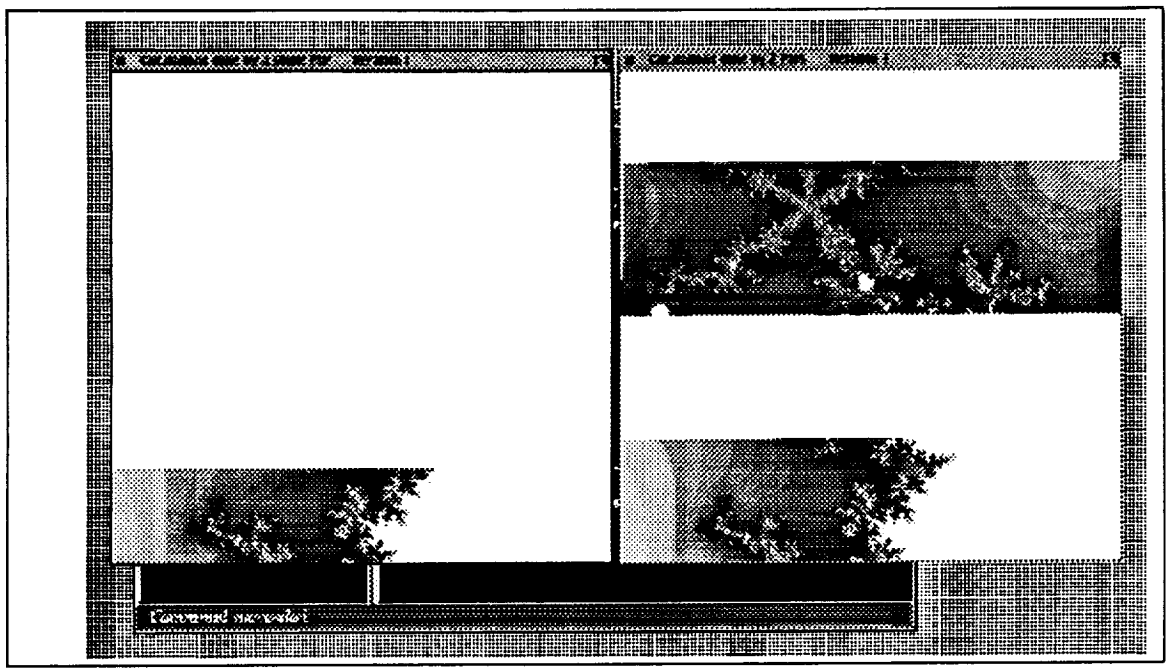

Figure 1: Running the demo

References

[1] R. A. Volz, R. Theriault, G. Smith, R. Waldrop, "Distributed and Parallel Execution in Ada 83," submitted to the Joint Workshop on Parallel and Distributed Real-Time Systems, Santa Barbara, California, April 1995.

[2] R. A. Volz, R. Theriault, R. Waldrop, S. J. Goldsack, A. Holzbacher-Valero, "Distributed and Parallel Ada and the Ada 9X Recommendations," accepted for *Distributed Systems Engineering Journal.*

[3] Ada 9X Mapping/Revision Team, "Programming Language Ada, Language and Standard Libraries," Intermetrics, Inc., Draft version 5.0, June 1994.

[4] S. J. Goldsack, A. Holzbacher-Valero, R. A. Volz, R. Waldrop, "AdaPT and Ada 9X," *Ada Letters*, vol. 14, no. 2, pp. 80-92, March/April 1994.

[5] A. Gargaro, S. Goldsack, A. Holzbacher-Valero, R. Volz, R. Waldrop, A. Wellings, "Supporting Distribution and Dynamic Reconfiguration in AdaPT," *Distributed Systems Engineering Journal*, vol. 1, no. 3, March 1994.

[6] R. Waldrop, R. A. Volz, G. W. Smith, A. Holzbacher-Valero, S. J. Goldsack, "On-Line Upgrade of Program Modules Using AdaPt," AIAA Computing in Aerospace 9 Conference, October 19-21, 1993, San Diego, CA.